ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP016

MIGRATING FROM ALARM HANDLER TO PHOEBUS ALARM-SERVER AT BESSY II AND HZB

M. Gotz*, T. Birke, Helmholtz-Zentrum Berlin, Berlin, Germany

Abstract

The BESSY II lightsource has been in operation at Helmholtz-Zentrum Berlin (HZB) for 25 years and is expected to be operated for more than the next decade. The EPICS Alarm Handler (ALH) has served as the basis for a reliable alarm system for BESSY II as well as other facilities and laboratories operated by HZB. To preempt software obsolescence and enable a centralized architecture for the alarm systems running throughout HZB, it is being migrated to the alarm-service developed within the Control System Studio/Phoebus ecosystem. To facilitate simultaneous operation of the old alarm system while evaluating the new system, tools were developed to automate creation of the Phoebus alarm-service configuration files in the control systems' build process. Additionally, tools and configurations were devised to mirror the old system's key features in the new one. This contribution presents the tools developed and the infrastructure deployed to use the Phoebus alarm-service at HZB.

MOTIVATION

The ALH [1] has served the HZB well as the basis for the current alarm system of the BESSY II light source for 25 years. However, with the prospect of operating BESSY II for another 10 years the desire arose to replace it with a more modern system. The key limitations of the ALH are that it is a single application requiring a graphical user interface (GUI) and that it is no longer actively maintained. Instances on different computers are mostly independent from one another, creating problems with synchronicity and requiring special care for automated actions which should only be performed once. Furthermore, no more active development means future obsolescence is always a risk and there will be no support for newer EPICS developments like pvAccess.

The Phoebus [2] alarm system uses an alarm server, of which only one instance is required, communicating with multiple GUIs via a Kafka message broker. This approach appealed greatly to us, eliminating most of the above mentioned shortcoming of the ALH. Furthermore, Phoebus seemed to us the only truly viable replacement candidate for ALH. There is, to our knowledge, no other alarm system in the EPICS ecosystem that is both actively developed and with contributions from more than one institute.

OVERVIEW AND SPECIFIC NEEDS

The Phoebus alarm system is well documented [2]. At its heart is a Kafka message broker. An alarm server reads the configuration from Kafka and writes alarm state changes to Kafka. Clients read alarm state changes from Kafka, display and allow changes to the configuration via Kafka. Additional services exist to log alarms to Elastic Search or log changes to the configuration.

The basic setup is simple enough. However, covering all our needs required additional developments. While manual addition of PVs to the alarm configuration of Phoebus is easy, at HZB we have a well established alarm configuration for BESSY II and our other labs. With roughly 10000 PVs in it, we needed an automated way to convert this configuration. Particularly, because parts of this configuration are generated from spreadsheets, which change fairly frequently.

Another aspect for us were access controls. The control system runs in a separate network, which should be able to operate in isolation. Also, changes to the alarm-system, like acknowledging alarms, should be possible from the entire network. Nevertheless, read-only remote access to the alarm-system should be possible from within the wider institute.

Finally, we required a method of notifying operators of alarms via an internal SMS-like service. For the ALH a script modified the configuration for one instance after deployment adding severity commands to each alarm. A more transparent system was desired for the new setup.

CONFIGURATION CONVERSION AND GENERATION

ALH and Phoebus offer different features in the alarm configuration. For instance, the force mask in ALH allows changes to latching or logging behavior, in addition to the ability to disable an alarm based on the value of other PVs. On the other hand, the annunciation is only present in Phoebus. Therefore, a perfect conversion between these two formats is not possible. Our goal was a tool to convert the unambiguous parts and inform us about the problematic portions. We developed the python package *phoebusalarm* [3] to accomplish this. It parses alh-files into a python object structure. From that structure both alh and phoebus xml-files can be exported. In addition, the package can be used to programmatically create an alarm tree and export it into either format. Using this tool and manual intervention where necessary, we converted our alarm tree into the phoebus format. After the manual conversion was complete the process was also integrated into our build process. This ensures any changes to alh-files remain forward compatible with phoebus. Additionally, all the resulting alarm-configuration files are combined into a single file and checked against a schema with xmllint ensuring the presence of all included files and syntactic correctness.

* malte.gotz@helmholtz-berlin.de

Content from this work may be used under the terms of the CC BY 4.0 licence (© 2023). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP016

SYSTEM LAYOUT

Where ALH is a single application, the Phoebus alarm system needs hat least a Kafka message broker, an alarm-server service and GUI Client(s). We needed to provision these services in a way that is ideally fault tolerant and satisfies our needs for access permissions. Those were that BESSYII operates on a segregated network that must be able to operate in isolation. All machines in the control-network should be able to acknowledge alarms. From outside the network we need to read the status of the alarm system. Furthermore, we want to add additional alarm servers from other networks in the future. The resulting layout is sketched in Fig. 1. The core idea is to have multiple Kafka clusters with mirroring. Inside the control network Kafka and the alarm-server run on virtual machines, providing the desired failover. Network restrictions limit access to these servers and the Kafka topics are writeable to anyone in the network. The Kafka Mirrormaker 2 functionality is used to mirror the cluster from inside the control network to another cluster outside. On the outside cluster, access control lists (ACLs) are setup to only allow the mirroring process write access. Otherwise, the topics are read only. Logging to an existing elastic search cluster and notifying operators is performed from this mirrored cluster.

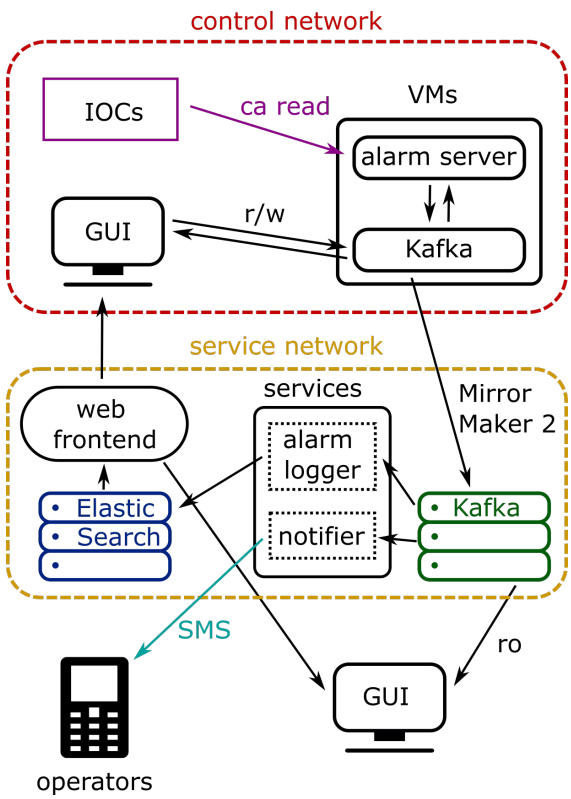

Figure 1: Layout of the components involved in the Phoebus alarm system at BESSYII. Core is the isolation of the control network and replication of the Kafka messages to an outside server, simplifying access management.

SENDING NOTIFICATIONS

Our existing setup uses an alh-configuration modified by a script to notify operators on every alarm. To replicate this behavior in Phoebus we would need to attach an automated action to every PV in the configuration. While this could be scripted, it is opaque and hard to modify, if, for instance, an alarm should not be notified on. On the other hand, Phoebus provides an annunciator feature as part of its alarm system. Every new alarm, for which annunciation is enabled, is written to a Kafka topic called <server-name>Talk by the alarm-server. Phoebus includes a Kafka-client that reads the topic and voices the alarm. We wrote our own Kafka-client, which instead of annunciating the alarms via speaker, sends out text notifications to the operators. The notification of operators is thus nicely controlled by the annunciation flag in the alarm configuration.

EXPERIENCES

The presented solutions have been implemented to a varying degree. An automatic conversion of the ALH-configuration to phoebus is part of our build process for over a year. It is stable and produces a faithful replication of the ALH-source. In addition, the automatic conversion combined with the xmllint run is a useful build-time check on the validity of the original configuration. Errors like missing or misspelled include files or duplicate group names were discovered und subsequently remedied, thereby improving the old system even before the new one is in production. Setup of the mirroring Kafka-clusters, with an alarm-server and logger is also completed and running in a test configuration for half a year. While the initial setup of a single node Kafka-cluster and an alarm-server is straight forward, going the a reliable system required more effort than initially anticipated. With the Phoebus alarm system we were required to operate and understand at least three software stacks: Phoebus, Kafka and Elasticsearch. Due to an existing Elasticsearch installation we only had two new systems, but this still created considerable work. Lacking previous experience with deploying java applications, even relatively simple tasks like limiting Kafka's log-output such that it would not exceed available disk-space, required several iterations. Bringing proof-of-concept installations into a reliable format to manage with Ansible was another multi stage process, that is in part still ongoing. Nevertheless, the test setup is working well. The replication across the clusters works as expected and access from inside and outside the control network is controlled without further configuration or authentication needs on the clients. Our own SMS-notification system as a Kafka-client is implemented, but is still awaiting larger testing. This is the final component still missing for us, before attempting to switch to Phoebus as our main system.

CONCLUSION

Migrating the BESSYII alarm system to Phoebus is well on its way and we could implement all our specific needs. The developed conversion tool might be useful for other

Content from this work may be used under the terms of the CC BY 4.0 licence (© 2023). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP016

sites atempting a simlar migration. However, converting an existing configuration is but the first step. Establishing reliable, manageable and scalable installation of the different services required for the Phoebus alarm-system has been the more time consuming task. Its difficulty heavily depends on existing infrastructure, experience and local requirements.

ACKNOWLEDGEMENTS

We thank the members of the accelerator control system group for their support. In particular, D. Engel, B. Franksen, V. Laux and S. Heise for their support in the setup of the various systems.

REFERENCES

[1] EPICS alarm handler, https://epics.anl.gov/extensions/alh/index.php

[2] Phoebus alarms, https://control-system-studio.readthedocs.io/en/latest/app/alarm/ui/doc/index.html

[3] M. Gotz, *Phoebusalarm*, version 2.1.2, 2023. https://github.com/hz-b/phoebusalarm

Content from this work may be used under the terms of the CC BY 4.0 licence (© 2023). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI